JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 7, ISSUE 09, 2020

Machine Learning based Phishing Email Detection

Akavaram Swapna, Mahesh Kumar Singirikonda, Prasanna Shivva

Department of Computer Science and Engineering

Sree Dattha Group of Institutions, Hyderabad, Telangana, India.

Abstract

Email has become one of the most important forms of communication. In 2014, there are estimated to be 4.1 billion email accounts worldwide, and about 196 billion emails are sent each day worldwide. Spam is one of the major threats posed to email users. In 2013, 69.6% of all email flows were spam. Links in spam emails may lead to users to websites with malware or phishing schemes, which can access and disrupt the receiver's computer system. These sites can also gather sensitive information from. Additionally, spam costs businesses around $2000 per employee per year due to decreased productivity. Therefore, an effective spam filtering technology is a significant contribution to the sustainability of the cyberspace and to our society. Current spam techniques could be paired with content-based spam filtering methods to increase effectiveness. Content-based methods analyze the content of the email to determine if the email is spam. The goal of our project was to analyze machine learning algorithms such as logistic regression, and naive bayes classifier algorithm and determine their effectiveness as content-based spam filters.

Keywords: Phishing email, machine learning, spam filtering.

1. Introduction

The rapid development of Internet technologies has immensely changed on-line users' experience, while security issues are also getting more overwhelming. The current situation is that new threats may not only cause severe damage to customers' computers but also aim to steal their money and identity. Among these threats, phishing is a noteworthy one and is a criminal activity that uses social engineering and technology to steal a victim's identity data and account information. According to a report from the Anti-Phishing Working Group (APWG), the number of phishing detections in the first quarter of 2018 increased by 46% compared with the fourth quarter of 2017 [1]. According to the striking data, phishing has shown an apparent upward trend in recent years. Similarly, the harm caused by phishing can be imagined as well.

For phishing, the most widely used and influential mean is the phishing email. Phishing email refers to an attacker using a fake email to trick the recipient into returning information such as an account password to a designated recipient. Additionally, it may be used to trick recipients into entering special web pages, which are usually disguised as real web pages, such as a bank's web page, to convince users to enter sensitive information such as a credit card or bank card number and password. Although the attack of phishing email seems simple, its harm is immense. In the United States alone, phishing emails are expected to bring a loss of 500 million dollars per year [2]. According to the APWG, the number of phishing emails increased from 68,270 in 2014 to 106,421 in 2015, and the number of different phishing emails reported from January to June 2017 was approximately 100,000. In addition, Gartner's report notes that the number of users who have ever received phishing emails has reached a total of 109 billion. Microsoft analyzes and scans over 470 billion emails in Office 365 every month to find phishing and malware. From January to December 2018, the proportion of inbound emails that were phishing emails increased by 250%. Great harm and strong growth momentum have forced people to pay attention to phishing emails. Therefore, many detection methods for phishing emails have been proposed.

Various techniques for detecting phishing emails are mentioned in the literature. In the entire technology development process, there are mainly three types of technical methods including blacklist mechanisms, classification algorithms based on machine learning and based on deep learning. From previous work, the existing detection methods based on the blacklist mechanism mainly rely on people's identification and reporting of phishing links requiring a large amount of manpower and time. However, applying artificial intelligence (AI) to the detection method based on a machine learning classification algorithm requires feature engineering to manually find representative features that are not conducive to the migration of application scenarios. Moreover, the current detection method based on deep learning is limited to word embedding in the content representation of the email. These methods directly transferred natural language processing (NLP) and deep learning technology, ignoring the specificity of phishing email detection so that the results were not ideal [3], [4].

Given the methods mentioned above and the corresponding problems, we set to study phishing email detection systematically based on deep learning. Specifically, this paper makes the following contributions:

1) With respect to the particularity of the email text, we analyze the email structure, and mine the text features from four more detailed parts: the email header, the email body, the word-level, and the char-level.
2) The RCNN model is improved by using the Bidirectional Long Short-Term Memory (Bi-LSTM). Then, the email is modelled from multiple levels using an improved RCNN model. Noise is introduced as little as possible, and the context information of the email can be better captured.
3) The attention mechanism is applied between the email header and the email body, and different weights are respectively assigned to the two parts so that the model can focus on more different and more useful information from the email header and the email body.
4) The THEMIS model proposed in this paper performs well on an unbalanced dataset. The accuracy achieves 99.848%, and all evaluation metrics of THEMIS are superior to the existing detection technologies.

2. Literature survey

Gangavarapu et al. [5] aimed at elucidated on the way of extracting email content and behavior-based features, what features are appropriate in the detection of UBEs, and the selection of the most discriminating feature set. Furthermore, to accurately handle the menace of UBEs, this work facilitated an exhaustive comparative study using several state-of-the-art machine learning algorithms. This proposed model resulted in an overall accuracy of 99% in the classification of UBEs. The text is accompanied by snippets of Python code, to enable the reader to implement the approaches elucidated in this paper.

Srinivasan et al. [6] presented a new methodology for detecting spam emails based on deep learning architectures in the context of natural language processing (NLP). Past works on classical machine learning based spam email detection has relied on various feature engineering methods. This proposed method leveraged the text representation of NLP and map towards spam email detection task. Various email representation methods are utilized to transform emails into email word vectors, as an essential step for machine learning algorithms. Moreover, optimal parameters are identified for many deep learning architectures and email representation by following the hyper-parameter tuning approach.

The performance of many classical machine learning classifiers and deep learning architectures with various text representations are evaluated based on publicly available three email corpora.

AbdulNabi et al. [7] introduced the effectiveness of word embedding in classifying spam emails. Pre-trained transformer model BERT (Bidirectional Encoder Representations from Transformers) is fine-tuned to execute the task of detecting spam emails from non-spam (HAM). BERT uses attention layers to take the context of the text into its perspective. Results are compared to a baseline DNN (deep neural network) model that contains a BiLSTM (bidirectional Long Short-Term Memory) layer and two stacked Dense layers. In addition, results are compared to a set of classic classifiers k-NN (k-nearest neighbors) and NB (Naive Bayes). Two open-source data sets are used, one to train the model and the other to test the persistence and robustness of the model against unseen data. The proposed approach attained the highest accuracy of 98.67% and 98.66% F1 score.

Alam et al. [8] developed a model to detect the phishing attacks using machine learning (ML) algorithms like random forest (RF) and decision tree (DT). A standard legitimate dataset of phishing attacks from Kaggle was aided for ML processing. To analyze the attributes of the dataset, the proposed model has used feature selection algorithms like principal component analysis (PCA). Finally, a maximum accuracy of 97% was achieved through the random forest algorithm.

Hassanpour et al. [9] presented some of the early results on the classification of spam email using deep learning and machine methods. This work utilized word2vec to represent emails instead of using the popular keyword or other rule-based methods. Vector representations are then fed into a neural network to create a learning model. This work has tested our method on an open dataset and found over 96% accuracy levels with the deep learning classification methods in comparison to the standard machine learning algorithms.

Kumar et al. [10] discussed the machine learning algorithms and applied all these algorithm on this data sets and best algorithm is selected for the email spam detection having best precision and accuracy.

3. Proposed system

3.1 Phishing email dataset

Phishing is the fraudulent attempt to obtain sensitive information or data, such as usernames, passwords, and credit card details, by disguising oneself as a trustworthy entity in an electronic communication. Typically carried out by email spoofing, instant messaging, and text messaging, phishing often directs users to enter personal information at a fake website which matches the look and feel of the legitimate site. Phishing is an example of social engineering techniques used to deceive users. Users are lured by communications purporting to be from trusted parties such as social web sites, auction sites, banks, colleagues/executives, online payment processors or IT administrators. In this challenge our goal is to train a classifier that will detect phishing emails.

Data fields

- index - Email Id
- Subject - The Email's headline
- Content - The Email's internal message
- Content-Type - The Email's content format

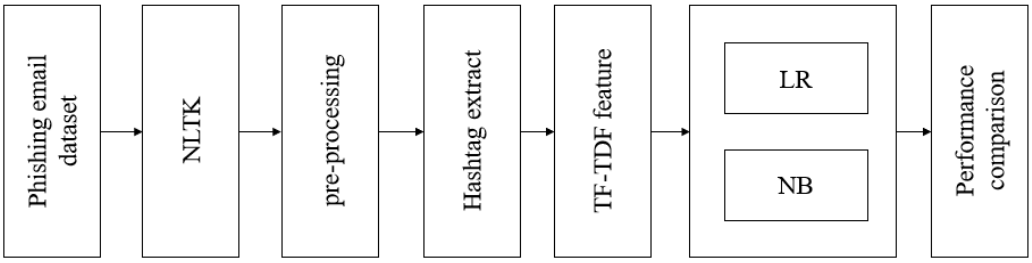


Fig. 1: Block diagram of proposed system.

3.2 Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project. NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

3.3 Data Preprocessing in Machine learning

Data Pre-processing in Machine learning

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

Why do we need Data Pre-processing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

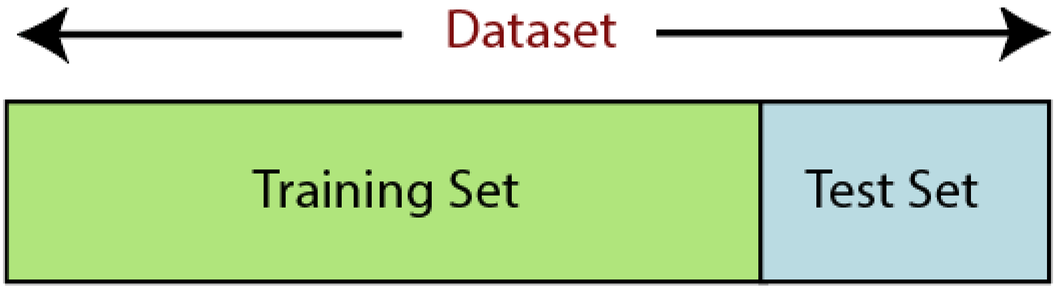
- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

3.3.1 Splitting the Dataset into the Training set and Test set

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.

Supposeif we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So, we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:



Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

3.4 TF-IDF Feature extraction

TF-IDF which stands for Term Frequency – Inverse Document Frequency. It is one of the most important techniques used for information retrieval to represent how important a specific word or phrase is to a given document. Let’s take an example, we have a string or Bag of Words (BOW) and we have to extract information from it, then we can use this approach.

The tf-idf value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general. TF-IDF use two statistical methods, first is Term Frequency and the other is Inverse Document Frequency. Term frequency refers to the total number of times a given term t appears in the document doc against (per) the total number of all words in the document and the inverse document frequency measure of how much information the word provides. It measures the weight of a given word in the entire document. IDF show how common or rare a given word is across all documents. TF-IDF can be computed as tf * idf

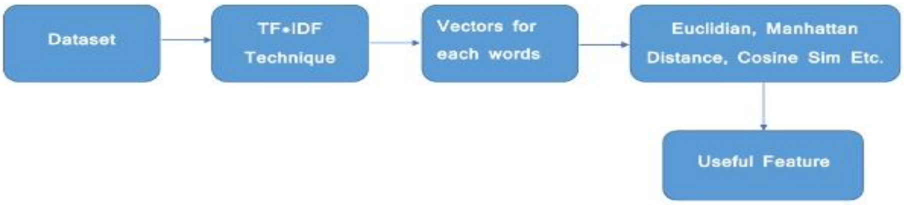


Fig. 2: TF-IDF block diagram.

TF-IDF do not convert directly raw data into useful features. Firstly, it converts raw strings or dataset into vectors and each word has its own vector. Then we'll use a particular technique for retrieving the feature like Cosine Similarity which works on vectors, etc.

Terminology:

t — term (word)

d — document (set of words)

N — count of corpus

corpus — the total document set

Term Frequency (TF): Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, "Data Science is awesome!" A simple way to start out is by eliminating documents that do not contain all three words "Data" is", "Science", and "awesome", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency. The weight of a term that occurs in a document is simply proportional to the term frequency.

$$tf(t,d) = count\ of\ t\ in\ d\ /\ number\ of\ words\ in\ d$$

Document Frequency: This measures the importance of document in whole set of corpus, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, whereas DF is the count of occurrences of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

$$df(t) = occurrence\ of\ t\ in\ documents$$

Inverse Document Frequency (IDF): While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. **The** IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (because stop words such as "is" is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

$$idf(t) = N/df$$

Now there are few other problems with the IDF, in case of a large corpus,say 100,000,000 , the IDF value explodes , to avoid the effect we take the log of idf . During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator.

$$idf(t) = log(N/(df + 1))$$

The TF-IDF now is at the right measure to evaluate how important a word is to a document in a collection or corpus. Here are many different variations of TF-IDF but for now let us concentrate on this basic version.

$$tf - idf(t,d) = tf(t,d) * log(N/(df + 1))$$

Implementing TF-IDF: To make TF-IDF from scratch in python, let's imagine those two sentences from different document:

first_sentence: "Data Science is the sexiest job of the 21st century".

second_sentence: "machine learning is the key for data science".

First step we have to create the TF function to calculate total word frequency for all documents.

3.5 Multinominal Naive Bayes Model

What is the Multinomial Naive Bayes algorithm?

Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature. The presence or absence of a feature does not affect the presence or absence of the other feature.

Applications

Naive Bayes algorithm is used in the following places:

- Face recognition
- Weather prediction
- Medical diagnosis
- Spam detection
- Age/gender identification
- Language identification
- Sentimental analysis
- Authorship identification
- News classification

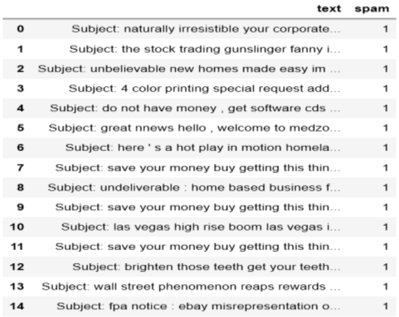
Advantages

The Naive Bayes algorithm has the following advantages:

- It is easy to implement as you only must calculate probability.
- You can use this algorithm on both continuous and discrete data.
- It is simple and can be used for predicting real-time applications.
- It is highly scalable and can easily handle large datasets.

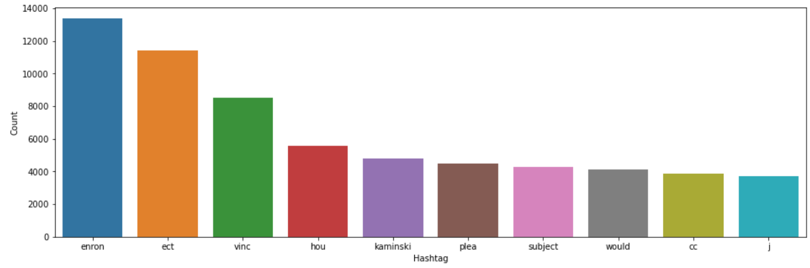
4. Results

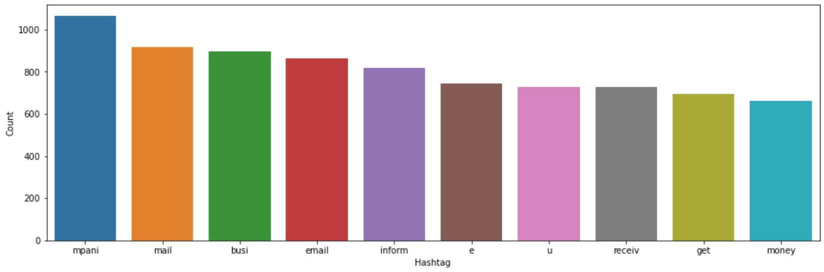
Sample dataset



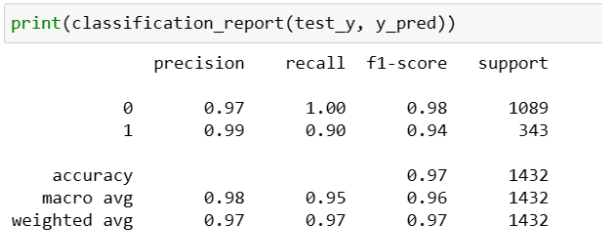
	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1
5	Subject: great nnews hello , welcome to medzo...	1
6	Subject: here ' s a hot play in motion homela...	1
7	Subject: save your money buy getting this thin...	1
8	Subject: undeliverable : home based business f...	1
9	Subject: save your money buy getting this thin...	1
10	Subject: las vegas high rise boom las vegas i...	1
11	Subject: save your money buy getting this thin...	1
12	Subject: brighten those teeth get your teeth...	1
13	Subject: wall street phenomenon reaps rewards ...	1
14	Subject: fpa notice : ebay misrepresentation o...	1

Data visualization

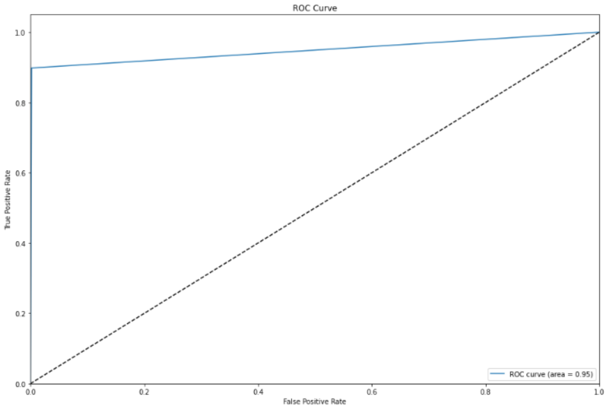


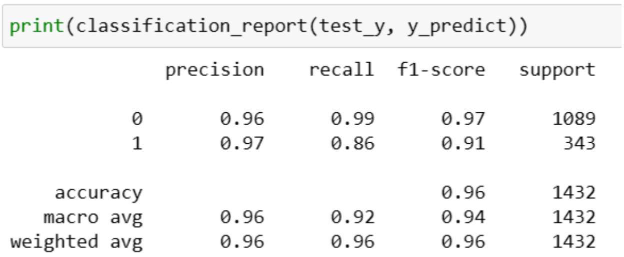


Logistic Regression Classification report

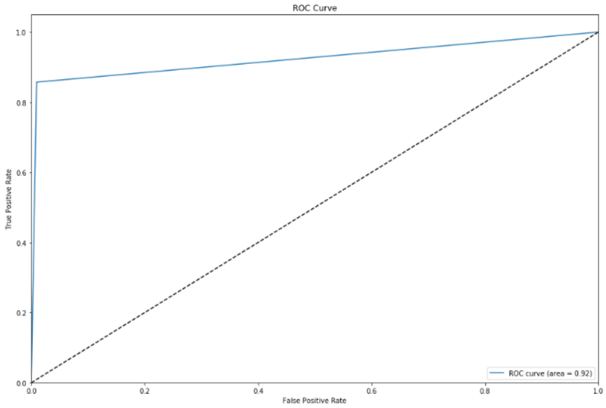


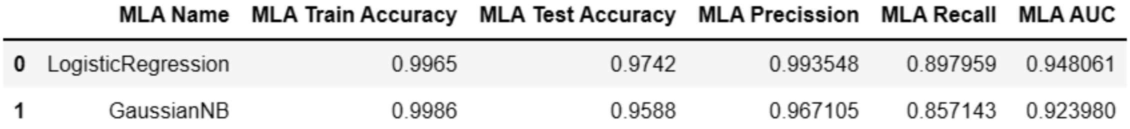
```
print(classification_report(test_y, y_pred))

              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1089
           1       0.99      0.90      0.94       343

    accuracy                           0.97      1432
   macro avg       0.98      0.95      0.96      1432
weighted avg       0.97      0.97      0.97      1432
```



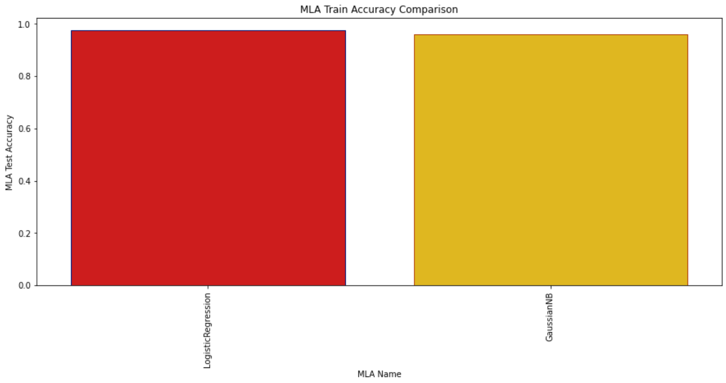


```
print(classification_report(test_y, y_predict))

              precision    recall  f1-score   support

           0       0.96      0.99      0.97      1089
           1       0.97      0.86      0.91       343

    accuracy                           0.96      1432
   macro avg       0.96      0.92      0.94      1432
weighted avg       0.96      0.96      0.96      1432
```





	MLA Name	MLA Train Accuracy	MLA Test Accuracy	MLA Precission	MLA Recall	MLA AUC
0	LogisticRegression	0.9965	0.9742	0.993548	0.897959	0.948061
1	GaussianNB	0.9986	0.9588	0.967105	0.857143	0.923980



5. Conclusion

A Machine Learning model has been developed to detect phishing attacks, which have been already sent to the user's mailbox. To accomplish this task, a systematic literature review (SLR) has been conducted to identify ML techniques with greater acceptance by researchers. The SLR allowed us to establish the most common parameters that evidenced that the mail has been infected with phishing. With such input, Naïve Bayes has been selected for the training stage and Logistic regression for the detection stage. To obtain this model, the phases of the typical ML modeling cycle have been used. The obtained result by using Naive Bayes and Logistic Regression has been of about 99% accuracy in the prediction of such attacks.

5.1 Future scope

As future work, we have planned to redesign this model using unsupervised learning techniques such as Deep Learning, to determine a greater number of features of an email with phishing to increase its detection accuracy. A residual neural network (ResNet) is an artificial neural network (ANN). It is a gateless or open-gated variant of the HighwayNet, the first working very deep feedforward neural network with hundreds of layers, much deeper than previous neural networks. Skip connections or shortcuts are used to jump over some layers (HighwayNets may also learn the skip weights themselves through an additional weight matrix for their gates). Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between. Models with several parallel skips are referred to as DenseNets. In the context of residual neural networks, a non-residual network may be described as a plain network.

References

[1] Anti-Phishing Working Group. (2018). Phishing Activity Trends Report 1st Quarter 2018. [Online]. Available: http://docs.apwg.org/ Preports/apwg_trends_report_q1_2018.pdf.

[2] M. Nguyen, T. Nguyen, and T. H. Nguyen. (2018). ‘‘A deep learning model with hierarchical LSTMs and supervised attention for anti-phishing.’’ [Online]. Available: https://arxiv.org/abs/1805.01554.

[3] M. Hiransha, N. A. Unnithan, R. Vinayakumar, and K. Soman, ‘‘Deep learning-based phishing e-mail detection,’’ in Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal. (IWSPA), A. D. R. Verma, Ed. Tempe, AZ, USA, Mar. 2018.

[4] C. Coyotes, V. S. Mohan, J. Naveen, R. Vinayakumar, and K. P. Soman, ‘‘ARES: Automatic rogue email spotter,’’ in Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal. (IWSPA), A. D. R. Verma, Ed. Tempe, AZ, USA, Mar. 2018.

[5] Gangavarapu, T., Jaidhar, C.D. & Chanduka, B. Applicability of machine learning in spam and phishing email filtering: review and approaches. Artif Intell Rev 53, 5019–5081 (2020). https://doi.org/10.1007/s10462-020-09814-9

[6] Srinivasan, S., Ravi, V., Alazab, M., Ketha, S., Al-Zoubi, A.M., Kotti Padannayil, S. (2021). Spam Emails Detection Based on Distributed Word Embedding with Deep Learning. In: Maleh, Y., Shojafar, M., Alazab, M., Baddi, Y. (eds) Machine Intelligence and Big Data Analytics for Cybersecurity Applications. Studies in Computational Intelligence, vol 919. Springer, Cham. https://doi.org/10.1007/978-3-030-57024-8_7.

[7] I AbdulNabi, Q. Yaseen, “Spam Email Detection Using Deep Learning Techniques”, Procedia Computer Science, Volume 184, 2021, Pages 853-858, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.03.107.

[8] M. N. Alam, D. Sarma, F. F. Lima, I. Saha, R. -E. -. Ulfath and S. Hossain, "Phishing Attacks Detection using Machine Learning Approach," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), 2020, pp. 1173-1179, doi: 10.1109/ICSSIT48917.2020.9214225.

[9] R. Hassanpour, E. Dogdu, R. Choupani, O. Goker, and N. Nazli. 2018. Phishing e-mail detection by using deep learning algorithms. In Proceedings of the ACMSE 2018 Conference (ACMSE '18). Association for Computing Machinery, New York, NY, USA, Article 45, 1. https://doi.org/10.1145/3190645.3190719.

[10] N. Kumar, S. Sonowal and Nishant, "Email Spam Detection Using Machine Learning Algorithms," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 108-113, doi: 10.1109/ICIRCA48905.2020.9183098.